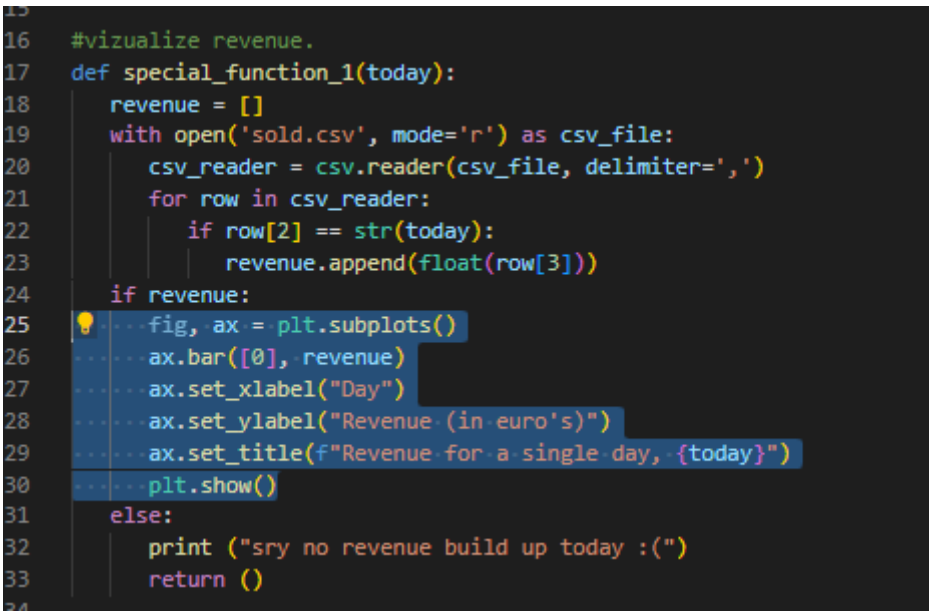
# Superpy.py assignment:

For the last two special functions I wrote a somewhat similar functions, because I wanted to work with Matplot library but lets dive deeper in the 5 highlighted lines.

1) On line 25, we create a figure and axis object that we can use to add elements to the graph.
2) On line 26, we create a bar graph with [0] being our x-coordinate and revenue being our height
3) On line 27, we set the label for the x-axis to “Day” because we only need to measure revenue of 1 single day.
4) On line 28, we set the label for the y-axis to “revenue (in euro’s)” because want to see how high the revenue goes on the giving date.
5) On line 29, we give the graph a title, I made it a f-string so if they decide to save it for a presentation later they have a indication of what day it was about.
6) On line 30, we display the graph on the screen.



```
16  #vizualize revenue.
17  def special_function_1(today):
18      revenue = []
19      with open('sold.csv', mode='r') as csv_file:
20          csv_reader = csv.reader(csv_file, delimiter=',')
21          for row in csv_reader:
22              if row[2] == str(today):
23                  revenue.append(float(row[3]))
24      if revenue:
25          fig, ax = plt.subplots()
26          ax.bar([0], revenue)
27          ax.set_xlabel("Day")
28          ax.set_ylabel("Revenue (in euro's)")
29          ax.set_title(f"Revenue for a single day, {today}")
30          plt.show()
31      else:
32          print ("sry no revenue build up today :(")
33          return ()
```

The three highlighted rows solve a lot in my code, it basically functions as follow:

1) We open the file sold.csv, in read mode. And we call that complete action csv_file.
2) Then we give a name, csv_reader to an action that returns a reader object wich wil iterate over the lines in the csv_file. You could say that the delimiter (‘,’) here acts like a border and will put its returned list of strings into a few elements.
3) After this we can use the elements to loop though.

```
#reports the revenue of a day.
def report_revenue(the_day):
    revenue = 0
    with open('sold.csv', mode='r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        for row in csv_reader:
            if row[2] == the_day:
                revenue += float(row[3])
    sys.stdout.write("\nin therms of revenue we made: " + str(revenue))
    return revenue
```

For the export_data_to_csv function I wrote a if statement that checks if the input from the user is a 4 digit number and nothing else. In some accounting programs you want to filter trough data and in this way the supermarket is kind of forced to keep a clean inventory of product numbers. It does how ever mean that the max number of products would be 10.000 from 0000 to 9999. But luckily This problem is easy to solve by raising the possible digits.

```
while args.ebp:
    ebp = input("Please enter the product id (must be a 4-digit number): ")
    if ebp.isdigit() and len(ebp) == 4:
        print("Input accepted.")
        export_data_to_CSV(ebp)
        break
    else:
        print("Invalid input")
        break
```